Introduction to Grammars

Mattox Beckman

beckman@iit.edu

Illinois Institute of Technology

Objectives

Your goal for this lecture is to learn how to do the following things:

- Identify and explain the parts of a grammar.
- Define *terminal*, *nonterminal*, *production*, *sentence*, *parse tree*, *left-recursive*, *ambiguous*.
- Use a grammar to draw the parse tree of a sentence.
- Identify a grammar that is *left-recursive*.
- Know about *ambiguous grammars*:
  - Be able to identify, demonstrate, and eliminate ambiguity.
- Define *First Set* and *Follow Set*.
- Compute the First Set from a grammar.
- Compute the Follow Set from a grammar.

Reminder: The Problem

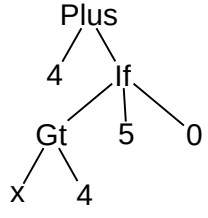
- Computer programs are entered as a stream of ASCII characters.

```
4 + if x > 4 then 5 else 0
```

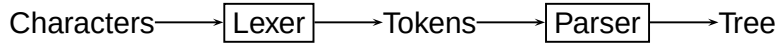
- We want to convert them into an *Abstract Syntax Tree*

ML code

```
1 PlusExp(
2   IntExp 4,
3   IfExp(
4     GtExp(VarExp "x",
5           IntExp 4),
6     IntExp 5,
7     IntExp 0))
```



Reminder: The Solution



The conversion from strings to trees is accomplished in two steps.

- First, convert the stream of characters into a stream of *tokens*.
  - This is called *lexing* or *scanning*.
  - Turns characters into words and categorizes them.
  - We did this in the last two lectures!
- Second, convert the stream of tokens into an abstract syntax tree.
  - This is called *parsing*.
  - Turns words into *sentences*.

What is in a sentence?

When we specify a sentence, we talk about two things that could be in them.

1. *Terminals*: tokens that are atomic — they have no smaller parts (e.g., “nouns”, “verbs”, “articles”)
2. *Non-terminals*: clauses that are not atomic — they are broken into smaller parts (e.g. “prepositional phrase”, “independent clause”, “predicate”)

Examples: (identify the terminals and the non-terminals)

- A sentence is a noun phrase, a verb, and a prepositional phrase
- A noun phrase is a determinant, and a noun
- A prepositional phrase is a preposition and a noun phrase.

Notation

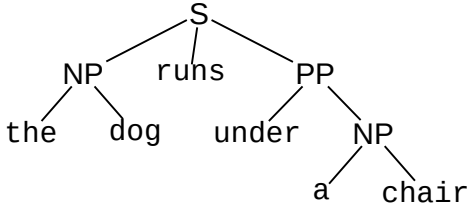
S → N verb P
N → det noun
P → prep N

- Each of the above lines is called a *production*.
  The *symbol* on the left hand side can be *produced* by collecting the symbols on the right hand side.
- The capital identifiers are *non-terminal* symbols.
- The lower case identifiers are *terminal* symbols.
- Because the left hand side is only a single non-terminal, the rules are *context free*. (Contrast: x S → NP verb PP)

Grammars specify trees...

“The dog runs under a chair.”

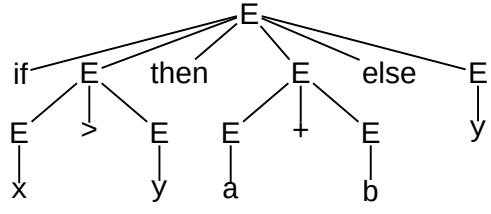
S → NP verb PP
NP → det noun
PP → prep NP



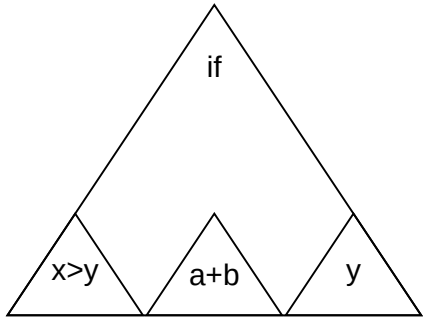
Another Example...

E → E + E
→ v
→ E > E
→ if E then E else E

```
if x > y then a + b else y
```



```
if x > y then a + b else y
```



It is important to be able to say what properties a grammar has.

Epsilon Productions A production of the form "E $\rightarrow \epsilon$", where ϵ represents the empty string.

Right Linear Grammars where all the productions have the form "E $\rightarrow$ x F" or "E $\rightarrow$ x".

Left-Recursive a production like "E $\rightarrow$ E + X"

Ambiguous More than one parse tree is possible for a specific sentence.

- Sometimes we want to specify that a symbol can become nothing.
- Example: "E $\rightarrow \epsilon$"
- Another example:

  S $\rightarrow$ NP verb PP

  NP$\rightarrow$ det A noun

  PP$\rightarrow$ prep NP

  A $\rightarrow$ adjective A

  A $\rightarrow \epsilon$

  This says that adjectives are an optional part of noun phrases.

- A *right linear* grammars is one in which all the productions have the form
  "E $\rightarrow$ x A" or "E $\rightarrow$ x".
- This corresponds to the *regular languages*.
- Example: regular expression `(10)*23` describes same language as this grammar:

  $A_0 \rightarrow$ 1A_1 | 2A_2

  $A_1 \rightarrow$ 0A_0

  $A_2 \rightarrow$ 3A_3

  $A_3 \rightarrow \epsilon$

Left-Recursive

- A grammar is *recursive* if the symbol being produced (the one on the left-hand side) also appears in the right hand side.

  Example: "E $\rightarrow$ if E then E else E"
- A grammar is *left-recursive* if the production symbol appears as the first symbol on the right-hand-side.

  Example: "E $\rightarrow$ E + F"
- ... or if is produced by a chain of left recursions ...

  Example: $A \rightarrow B$x
  $B \rightarrow A$y
- Question: why do we care if it's left-recursive?

Recursive Descent Parsers

- One way to use a grammar is to form a *recursive descent parser*.
- Start with a grammar and make some types ...

  S $\rightarrow$ NP verb PP
  NP$\rightarrow$ det noun
  PP$\rightarrow$ prep NP

```
1 type tree = S of (tree * token * tree)
2           | NP of (token * token)
3           | PP of (token * tree)
4 and token = Verb of string
5           | Det of string
6           | Noun of string
7           | Prep of string
```


Recursive Descent, II

- Next, write a function for each of the productions.
- Each function eats some terminal symbols and returns a pair...

```
1 let getS tlist =
2   let (np, newlist) = getNP tlist in
3   let (verb, newlist) = getVerb newlist in
4   let (pp, newlist) = getPP newlist in
5     S (np,verb,pp), newlist
6 and getVerb tlist = match (hd tlist) with
7   Verb x -> (Verb x, tl tlist)
8 | _ -> raise (Failure "parse error")
9 and getNP tlist = ...
```

- What would happen if we had a left-recursive rule?

Ambiguous Grammars

- A grammar is *ambiguous* if it can produce more than one parse tree for a single sentence.
- There are two common forms of ambiguity:
  - The "dangling else" form:

    E$\rightarrow$ if E then E else E
    E$\rightarrow$ if E then E
    E$\rightarrow$ whatever

    Example: `if a then if x then y else z` ... to which `if` does the `else` belong?
  - The "double-ended recursion" form: E$\rightarrow$ E + E
    E$\rightarrow$ E * E

    Example "3 + 4 * 5" ... is it "(3 + 4) * 5" or "3 + (4 * 5)"?

Fixing Ambiguity

- Ambiguity can often be eliminated by thinking more carefully about what you are trying to express with your grammar.
- "Dangling else" usually matches with the nearest `if`. This can be encoded in the grammar. See §4.3 of the Dragon Book for details.

Fixing Ambiguity

- The "double-ended recursion" form usually reveals a lack of precedence and associativity information. A technique called *stratification* often fixes this.
  - Left-recursive means "associates to the left", similarly right-recursive.
  - Higher precedence rules occur lower in the grammar.

E→ F + E
E→ F
F→ T * E
F→ T
T→ (E)
T→ integer

Try the following problems

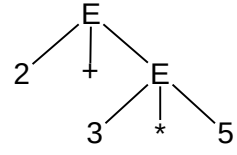
1. Draw two separate parse trees for the sentence `2 + 3 * 5` given the grammar
   E→ E + E
   E→ E * E
   E→ integer
2. Now draw a tree for the same sentence using the grammar
   E→ F + E F→ T * E T→ (E)
   E→ F F→ T T→ integer
3. Draw two separate parse trees for the "dangling else" example:

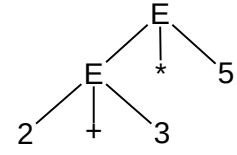
```
if a then if x then y else z
```

E→ if E then E else E
E→ if E then E
E→ var

Problem 1

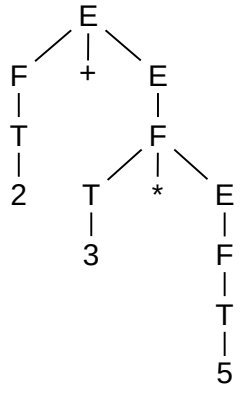
- Draw two separate parse trees for the sentence `2 + 3 * 5` given the grammar
  E→ E + E
  E→ E * E
  E→ integer





- Now draw a tree for the same sentence using the grammar
  - E→ F + E
  - E→ F
  - F→ T * E
  - F→ T
  - T→ (E)
  - T→ integer

These large tree sizes can cause our parser to be inefficient.

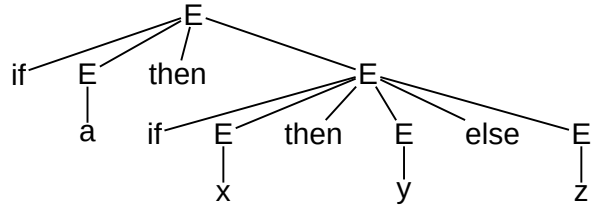


- Draw two separate parse trees for the "dangling else" example:

`if a then if x then y else z`

E→ if E then E else E
E→ if E then E
E→ var

Tree 1:

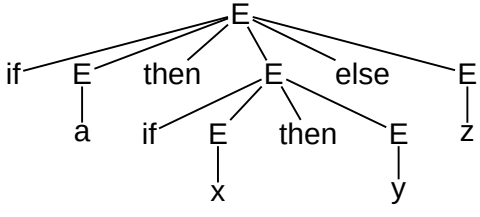


- Draw two separate parse trees for the "dangling else" example:

`if a then if x then y else z`

E→ if E then E else E
E→ if E then E
E→ var

Tree 2:



- Given a grammar for a language L, how can we recognize a sentence in L?
- Solution: Divide and Conquer: Given a symbol E...
  - What symbols indicate that the symbol E is just starting? (First Set)
  - What symbols should we expect to see after we have finished parsing an E?

Misleadingly simple example: $S \rightarrow xEy$ First(E) $=\{z, q\}$
$E \rightarrow zE$ Follow(E)=$\{y\}$
$E \rightarrow q$

- Important because a parser can see only a few tokens at once.

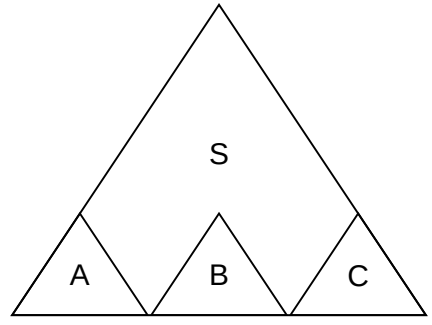
Algorithm

We can compute the FIRST set by a simple iterative algorithm. For each symbol X.

1. if X is a terminal, then $First(X) = \{X\}$
2. if there is a production $X \rightarrow \epsilon$, then add ϵ to $First(X)$.
3. if there is a production $X \rightarrow Y_1 Y_2 \cdots Y_n$, then add $First(Y_1 Y_2 \cdots Y_n)$ to $First(X)$:
   - If $First(Y_1)$ does not contain ϵ, then $First(Y_1 Y_2 \cdots Y_n) = First(Y_1)$.
   - Otherwise, $First(Y_1 Y_2 \cdots Y_n) = First(Y_1)/\epsilon \cup First(Y_2 \cdots Y_n)$
   - If all of $Y_1, Y_2, \ldots Y_n$ have ϵ then add ϵ to $First(X)$.

First Set Picture

$$S \rightarrow A\ B\ C$$



First Set Example

Consider the grammar

$S \rightarrow$ `if` E `then` S ;
$S \rightarrow$ `print` E;
$E \rightarrow E$ + E
$E \rightarrow P$ `id`
$P \rightarrow *\ P$
$P \rightarrow \epsilon$

Step 1: Create a list of symbols....

S={}
E={}
P={}

First Set Example

Consider the grammar

$S \rightarrow$ `if` E `then` S ; $\Leftarrow$
$S \rightarrow$ `print` E; $\Leftarrow$
$E \rightarrow E$ + E
$E \rightarrow P$ `id`
$P \rightarrow *\ P$ $\Leftarrow$
$P \rightarrow \epsilon$ $\Leftarrow$

Step 2: Add terminals starting productions, and all ϵ.

S={`if`, `print` }
E={}
P={ϵ, `*`}

Consider the grammar $S \rightarrow \texttt{if}\ E\ \texttt{then}\ S\ ;$
$S \rightarrow \texttt{print}\ E;$
$E \rightarrow E + E$
$E \rightarrow P\ \texttt{id}$ ⇐
$P \rightarrow * P$
$P \rightarrow \epsilon$

Step 3: Check productions. Add $First(P\texttt{id})$ to $First(E)$.

S={if, print }
E={*, id}
P={ϵ, *}

Introduction to Grammars
First Set Example
February 7, 2007
§5 **First Sets**

Consider the grammar $S \rightarrow \texttt{if}\ E\ \texttt{then}\ S\ ;$
$S \rightarrow \texttt{print}\ E;$
$E \rightarrow E + E$ ⇐
$E \rightarrow P\ \texttt{id}$
$P \rightarrow * P$
$P \rightarrow \epsilon$

Step 3: Check productions: $E \rightarrow E + E$ adds nothing.

S={if, print }
E={*, id}
P={ϵ, *}

Illinois Institute of Technology Mattox Beckman
Introduction to Grammars – p. 27

$S \rightarrow A\texttt{x}$
$S \rightarrow B\texttt{y}$
$S \rightarrow \texttt{z}$
$A \rightarrow \texttt{1}CB$
$A \rightarrow \texttt{2}B$
$B \rightarrow \texttt{3}B$
$B \rightarrow C$
$C \rightarrow \texttt{4}$
$C \rightarrow \epsilon$

Create a chart:
S={}
A={}
B={}
C={}

Introduction to Grammars
Another First Set Example
February 7, 2007
§5 **First Sets**

$S \rightarrow A\texttt{x}$
$S \rightarrow B\texttt{y}$
$S \rightarrow \texttt{z}$ ⇐
$A \rightarrow \texttt{1}CB$ ⇐
$A \rightarrow \texttt{2}B$ ⇐
$B \rightarrow \texttt{3}B$ ⇐
$B \rightarrow C$
$C \rightarrow \texttt{4}$ ⇐
$C \rightarrow \epsilon$ ⇐

Add initial terminals and ϵs.
S={z}
A={1, 2}
B={3}
C={ϵ, 4}

Illinois Institute of Technology Mattox Beckman
Introduction to Grammars – p. 28

$S \to A\mathtt{x} \Leftarrow$
$S \to B\mathtt{y}$
$S \to \mathtt{z}$
$A \to \mathtt{1}CB$
$A \to \mathtt{2}B$
$B \to \mathtt{3}B$
$B \to C$
$C \to \mathtt{4}$
$C \to \epsilon$

Add $First(A\mathtt{x})$ to $First(S)$.
S={z, 1, 2}
A={1, 2}
B={3}
C={ϵ, 4}

$S \to A\mathtt{x}$
$S \to B\mathtt{y} \Leftarrow$
$S \to \mathtt{z}$
$A \to \mathtt{1}CB$
$A \to \mathtt{2}B$
$B \to \mathtt{3}B$
$B \to C$
$C \to \mathtt{4}$
$C \to \epsilon$

Add $First(B\mathtt{y})$ to $First(S)$.
S={z, 1, 2, 3}
A={1, 2}
B={3}
C={ϵ, 4}
Note that there is still more to be added to $First(B)$! We will have to revisit this step later.

$S \to A\mathtt{x}$
$S \to B\mathtt{y}$
$S \to \mathtt{z}$
$A \to \mathtt{1}CB$
$A \to \mathtt{2}B$
$B \to \mathtt{3}B$
$B \to C \Leftarrow$
$C \to \mathtt{4}$
$C \to \epsilon$

Add $First(C)$ to $First(B)$.
S={z, 1, 2, 3}
A={1, 2}
B={3, 4, ϵ}
C={ϵ, 4}
At this point we should iterate again to see if anything changes.

$S \to A\mathtt{x} \Leftarrow$
$S \to B\mathtt{y}$
$S \to \mathtt{z}$
$A \to \mathtt{1}CB$
$A \to \mathtt{2}B$
$B \to \mathtt{3}B$
$B \to C$
$C \to \mathtt{4}$
$C \to \epsilon$

Add $First(A\mathtt{x})$ to $First(S)$ again.
S={z, 1, 2, 3}
A={1, 2}
B={3, 4, ϵ }
C={ϵ, 4}
Nothing happens...

Another First Set Example

$S \to A\mathtt{x}$
$S \to B\mathtt{y}$ ⇐
$S \to \mathtt{z}$
$A \to \mathtt{1}CB$
$A \to \mathtt{2}B$
$B \to \mathtt{3}B$
$B \to C$
$C \to \mathtt{4}$
$C \to \epsilon$

Add $First(B\mathtt{y})$ to $First(S)$ again.
S={z, 1, 2, 3, 4, y}
A={1, 2}
B={3, 4, ϵ }
C={ϵ, 4}
The 4 gets propagated.... also, since B could be ϵ we need to add y.

Another First Set Example

$S \to A\mathtt{x}$
$S \to B\mathtt{y}$
$S \to \mathtt{z}$
$A \to \mathtt{1}CB$
$A \to \mathtt{2}B$
$B \to \mathtt{3}B$
$B \to C$ ⇐
$C \to \mathtt{4}$
$C \to \epsilon$

Add $First(C)$ to $First(B)$ again.
S={z, 1, 2, 3, 4, y}
A={1, 2}
B={3, 4, ϵ }
C={ϵ, 4}
Nothing happens...
At this point, if we should iterate again.
But nothing happens when you do that, so we are done.

Follow Sets

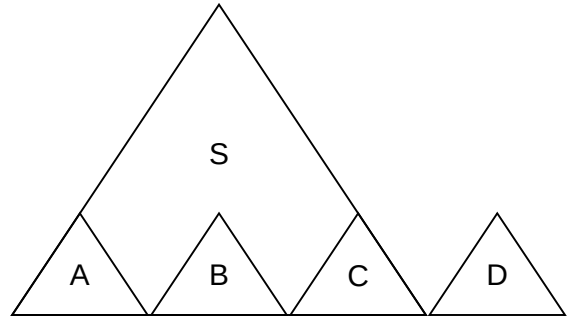
- Given a non-terminal symbol S, what terminal symbols could come after strings that are derived from S?

The Algorithm:

1. Put $ in $FOLLOW(S)$, where S is the start symbol.
   $ represents the “end of input.”
2. If there is a production $X \to \alpha Y \beta$, then add $FIRST(\beta)$ (but not ϵ) to $FOLLOW(Y)$.
3. If there is a production $X \to \alpha Y$, or if there is a production $X \to \alpha Y \beta$, where $\epsilon \in FIRST(\beta)$ then add $FOLLOW(X)$ to $FOLLOW(Y)$.

Follow Set Picture

$S \to A\ B\ C,$ and let D in Follow(S)



$S \rightarrow$ `if` E `then` S ;
$S \rightarrow$ `print` E;
$E \rightarrow E$ + E
$E \rightarrow P$ `id` P
$P \rightarrow *\ P$
$P \rightarrow \epsilon$

Make a chart, add $ to S.

S={$}
E={}
P={}

$S \rightarrow$ `if` E `then` S ; ⇐
$S \rightarrow$ `print` E;
$E \rightarrow E$ + E
$E \rightarrow P$ `id` P
$P \rightarrow *\ P$
$P \rightarrow \epsilon$

Check productions: add `then` to $FOLLOW(E)$,
and `;` to $FOLLOW(S)$

S={$, ;}
E={then}
P={}

$S \rightarrow$ `if` E `then` S ;
$S \rightarrow$ `print` E; ⇐
$E \rightarrow E$ + E ⇐
$E \rightarrow P$ `id` P
$P \rightarrow *\ P$
$P \rightarrow \epsilon$

Check productions: add `;` and `+` to $FOLLOW(E)$

S={$, ;}
E={then, ;, +}
P={}

$S \rightarrow$ `if` E `then` S ;
$S \rightarrow$ `print` E;
$E \rightarrow E$ + E
$E \rightarrow P$ `id` P ⇐
$P \rightarrow *\ P$
$P \rightarrow \epsilon$

Check productions: add `id` to $FOLLOW(P)$

S={$, ;}
E={then, ;, +}
P={id}

Follow Set Example

$S \rightarrow$ `if` E `then` S `;`
$S \rightarrow$ `print` E`;`
$E \rightarrow E$ `+` E
$E \rightarrow P$ `id` $P \Leftarrow$
$P \rightarrow *\ P$
$P \rightarrow \epsilon$

Check endings: P ends this rule,
so add $FOLLOW(E)$ to $FOLLOW(P)$.

S={$, ;}
E={then, ;, +}
P={id, then, ;, +}

Introduction to Grammars
February 7, 2007
§6 **Follow Sets**

Follow Set Example

$S \rightarrow$ `if` E `then` S `;`
$S \rightarrow$ `print` E`;`
$E \rightarrow E$ `+` E
$E \rightarrow P$ `id` P
$P \rightarrow *\ P$
$P \rightarrow \epsilon$

Done.

S={$, ;}
E={then, ;, +}
P={id, then, ;, +}

Another Follow Set Example

$S \rightarrow A$`x`
$S \rightarrow B$`y`
$S \rightarrow$ `z`
$A \rightarrow$ `1`CB
$A \rightarrow$ `2`B
$B \rightarrow$ `3`B
$B \rightarrow C$
$C \rightarrow$ `4`
$C \rightarrow \epsilon$

S={$}
A={}
B={}
C={}
Create a table, and add $ to $Follow(S)$.

Introduction to Grammars
February 7, 2007
§6 **Follow Sets**

Another Follow Set Example

$S \rightarrow A$`x` $\Leftarrow$
$S \rightarrow B$`y`
$S \rightarrow$ `z`
$A \rightarrow$ `1`CB
$A \rightarrow$ `2`B
$B \rightarrow$ `3`B
$B \rightarrow C$
$C \rightarrow$ `4`
$C \rightarrow \epsilon$

S={ $}
A={x}
B={}
C={}
Add x to $Follow(A)$.

$S \to A\mathtt{x}$
$S \to B\mathtt{y}$ ⇐
$S \to \mathtt{z}$
$A \to \mathtt{1}CB$
$A \to \mathtt{2}B$
$B \to \mathtt{3}B$
$B \to C$
$C \to \mathtt{4}$
$C \to \epsilon$

S={ $}
A={x}
B={y}
C={}
Add y to $Follow(B)$.

$S \to A\mathtt{x}$
$S \to B\mathtt{y}$
$S \to \mathtt{z}$ ⇐
$A \to \mathtt{1}CB$
$A \to \mathtt{2}B$
$B \to \mathtt{3}B$ ⇐
$B \to C$
$C \to \mathtt{4}$ ⇐
$C \to \epsilon$ ⇐

S={ $}
A={x}
B={y}
C={}
These productions add nothing.

$S \to A\mathtt{x}$
$S \to B\mathtt{y}$
$S \to \mathtt{z}$
$A \to \mathtt{1}CB$ ⇐
$A \to \mathtt{2}B$
$B \to \mathtt{3}B$
$B \to C$
$C \to \mathtt{4}$
$C \to \epsilon$

S={ $}
A={x}
B={y}
C={3, 4}
Add $First(B)$ to $Follow(C)$

$S \to A\mathtt{x}$
$S \to B\mathtt{y}$
$S \to \mathtt{z}$
$A \to \mathtt{1}CB$ ⇐
$A \to \mathtt{2}B$ ⇐
$B \to \mathtt{3}B$
$B \to C$
$C \to \mathtt{4}$
$C \to \epsilon$

S={ $}
A={x}
B={x, y}
C={3, 4}
Add $Follow(A)$ to $Follow(B)$.

$S \rightarrow A\texttt{x}$
$S \rightarrow B\texttt{y}$
$S \rightarrow \texttt{z}$
$A \rightarrow \texttt{1}CB \Leftarrow$
$A \rightarrow \texttt{2}B$
$B \rightarrow \texttt{3}B$
$B \rightarrow C$
$C \rightarrow \texttt{4}$
$C \rightarrow \epsilon$

S={ $}
A={x}
B={x, y}
C={x, 3, 4}

B can become ϵ, so add $Follow(A)$ to $Follow(C)$.

$S \rightarrow A\texttt{x}$
$S \rightarrow B\texttt{y}$
$S \rightarrow \texttt{z}$
$A \rightarrow \texttt{1}CB$
$A \rightarrow \texttt{2}B$
$B \rightarrow \texttt{3}B$
$B \rightarrow C \Leftarrow$
$C \rightarrow \texttt{4}$
$C \rightarrow \epsilon$

S={ $}
A={x}
B={x, y}
C={x, y, 3, 4}

Add $Follow(B)$ to $Follow(C)$. Now we're done.

Compute the First and Follow sets for this grammar.

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E'$
$E' \rightarrow \epsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T'$
$T' \rightarrow \epsilon$
$F \rightarrow (\ E\ )$
$F \rightarrow \texttt{id}$

Shorthand notation:

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E' \mid \epsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T' \mid \epsilon$
$F \rightarrow (\ E\ ) \mid \texttt{id}$

Step 1: Create the chart.

$E \rightarrow T\ E'$ E ={}
$E' \rightarrow +\ T\ E' \mid \epsilon$ E'={}
$T \rightarrow F\ T'$ T ={}
$T' \rightarrow *\ F\ T' \mid \epsilon$ T'={}
$F \rightarrow (\ E\ ) \mid \texttt{id}$ F ={}

Step 2: Add terminals and epsilons.

$E \rightarrow T\ E'$	E ={}
$E' \rightarrow +\ T\ E' \mid \epsilon$	E'={+, ϵ}
$T \rightarrow F\ T'$	T ={}
$T' \rightarrow *\ F\ T' \mid \epsilon$	T'={*, ϵ}
$F \rightarrow (\ E\ ) \mid$ `id`	F ={(, `id`}

Step 3: T starts with F; so add F entries to T.

$E \rightarrow T\ E'$	E ={}
$E' \rightarrow +\ T\ E' \mid \epsilon$	E'={+, ϵ}
$T \rightarrow F\ T'$	T ={(, `id`}
$T' \rightarrow *\ F\ T' \mid \epsilon$	T'={*, ϵ}
$F \rightarrow (\ E\ ) \mid$ `id`	F ={(, `id`}

Step 4: E starts with T; so add T entries to E.

$E \rightarrow T\ E'$	E ={(, `id`}
$E' \rightarrow +\ T\ E' \mid \epsilon$	E'={+, ϵ}
$T \rightarrow F\ T'$	T ={(, `id`}
$T' \rightarrow *\ F\ T' \mid \epsilon$	T'={*, ϵ}
$F \rightarrow (\ E\ ) \mid$ `id`	F ={(, `id`}

$E \rightarrow T\ E'$	E ={(, `id`}
$E' \rightarrow +\ T\ E' \mid \epsilon$	E'={+, ϵ}
$T \rightarrow F\ T'$	T ={(, `id`}
$T' \rightarrow *\ F\ T' \mid \epsilon$	T'={*, ϵ}
$F \rightarrow (\ E\ ) \mid$ `id`	F ={(, `id`}

Create a chart, add $ to S.

$E \rightarrow T\ E'$	E ={ $}
$E' \rightarrow +\ T\ E' \mid \epsilon$	E'={}
$T \rightarrow F\ T'$	T ={}
$T' \rightarrow *\ F\ T' \mid \epsilon$	T'={}
$F \rightarrow (\ E\ ) \mid$ `id`	F ={}

Add) to E.

$E \rightarrow T\ E'$	E ={), $ }
$E' \rightarrow +\ T\ E' \mid \epsilon$	E'={}
$T \rightarrow F\ T'$	T ={}
$T' \rightarrow *\ F\ T' \mid \epsilon$	T'={}
$F \rightarrow (\ E\ ) \mid$ `id`	F ={}

Add E to E'.

$E \rightarrow T\ E'$	E ={), $ }
$E' \rightarrow +\ T\ E' \mid \epsilon$	E'={), $ }
$T \rightarrow F\ T'$	T ={}
$T' \rightarrow *\ F\ T' \mid \epsilon$	T'={}
$F \rightarrow (\ E\ ) \mid$ `id`	F ={}

E' could be ϵ, so Add E to T.

$E \rightarrow T\ E'$	E ={), $ }
$E' \rightarrow +\ T\ E' \mid \epsilon$	E'={), $ }
$T \rightarrow F\ T'$	T ={), $ }
$T' \rightarrow *\ F\ T' \mid \epsilon$	T'={}
$F \rightarrow (\ E\ ) \mid$ `id`	F ={}

Follow Set Answer

E' follows T, so add $First(E')$ to T.

$E \rightarrow T\ E'$	E ={), $ }
$E' \rightarrow +\ T\ E' \mid \epsilon$	E'={), $ }
$T \rightarrow F\ T'$	T ={+,), $ }
$T' \rightarrow *\ F\ T' \mid \epsilon$	T'={}
$F \rightarrow (\ E\ ) \mid$ `id`	F ={}

Follow Set Answer

T ends with T', so add T to T'.

$E \rightarrow T\ E'$	E ={), $ }
$E' \rightarrow +\ T\ E' \mid \epsilon$	E'={), $ }
$T \rightarrow F\ T'$	T ={+,), $ }
$T' \rightarrow *\ F\ T' \mid \epsilon$	T'={+,), $ }
$F \rightarrow (\ E\ ) \mid$ `id`	F ={}

Follow Set Answer

T' could be ϵ, so add T to F.

$E \rightarrow T\ E'$	E ={), $ }
$E' \rightarrow +\ T\ E' \mid \epsilon$	E'={), $ }
$T \rightarrow F\ T'$	T ={+,), $ }
$T' \rightarrow *\ F\ T' \mid \epsilon$	T'={+,), $ }
$F \rightarrow (\ E\ ) \mid$ `id`	F ={+,), $ }

Follow Set Answer

T' follows F, so add $First(T')$ to F.

$E \rightarrow T\ E'$	E ={), $ }
$E' \rightarrow +\ T\ E' \mid \epsilon$	E'={), $ }
$T \rightarrow F\ T'$	T ={+,), $ }
$T' \rightarrow *\ F\ T' \mid \epsilon$	T'={+,), $ }
$F \rightarrow (\ E\ ) \mid$ `id`	F ={+, *,), $ }

... and now we're done.